CAPITOLIS

Take home assignment

Overview

This assignment is designed to give you an idea of the work you'll be facing day-to-day in this role, while also providing you the opportunity to showcase your thought process and skill set. Your solution will never be used for any purposes beyond evaluation.

We want you to build an app which is able to compress transactions.

What is a compression? I owe you $400, you owe me $500, we both have to reserve capital against this which we can’t use elsewhere! Let’s change that to me owing you $0 and you owing me $100. Why is this better? both of us just freed 400$ of capital.

For example:

```
const transactions = [
  { tradingParty: "me", counterparty: "you", amount: -400 },
  { tradingParty: "me", counterparty: "you", amount: 500 },
  { tradingParty: "me", counterparty: "someone_else", amount: 100 },
]

const compressedTransactions = compress(transactions);
/*
compressedTransactions = [
  { tradingParty: "me", counterparty: "you", amount: 100 },
  { tradingParty: "me", counterparty: "someone_else", amount: 100 },
];
*/
```

App requirements

- We'll need a basic interface to add and view transactions.
- A valid transaction has an amount and two parties: trading party and counterparty.
- The app should be able to support these capabilities:
  - Display 2 lists of all transactions grouped by:
    - Paying transactions (amount < 0).
    - Receiving transactions (amount > 0).
  - Add a new transaction:
    - A modal or dialog should open once `Add new Transaction` is clicked.
    - Only valid transactions should be allowed:
      - Counterparty should be a non empty string. Amount is a non-zero positive integer.
      - For the sake of this exercise the app should assume the trading party is always you, so `tradingParty = "me"`.

    - Once a transaction is added, it cannot be deleted.
- Once `Compress Transactions` is clicked the compression algorithm should be executed for all transactions. The resulting transactions should be automatically downloaded to a CSV file.

Guidelines

- For easy debugging the server should print details about incoming requests and outgoing responses.
- Transactions should be available as long as the server is up.
- We've added a mock design to help visualize how the app should look like, feel free to add your personal touch to the final design.
- We are primarily intersted in the quality and efficiency of the code. It should be a first draft that you are proud of.
- Please use React and NodeJS to complete this assignment.

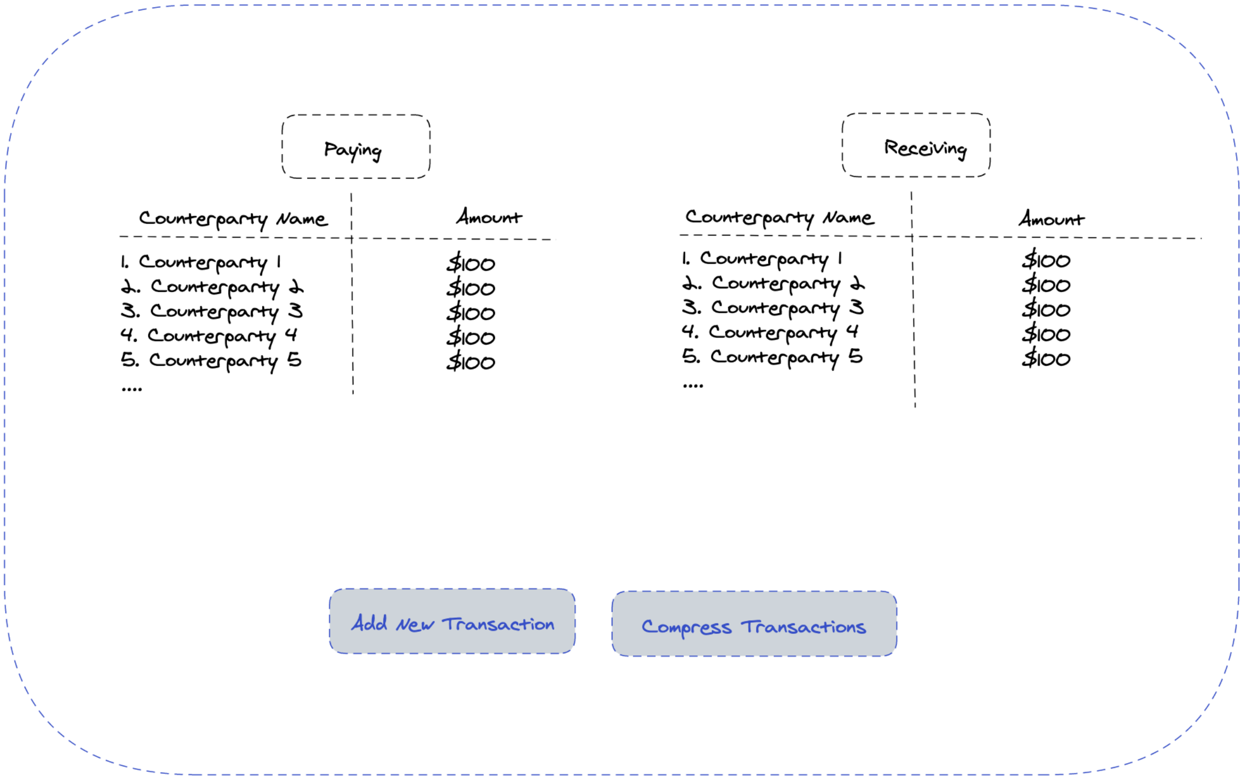
Bonuses

- Add some tests where you see fit.
- More complex persistency - read/write to JSON file, DB etc...
- Add more features you think are a good fit. For example:
    - Deploy the app to the web.
    - Add the ability to add transactions for which the `tradingParty` isn't you.
- Introduce a concept of identity and authorization to the app:
    - Basic login page with only user ID and no password.
    - In memory Sessions.
    - Transactions can be added by admins only
- Responsive design.

Evaluation

Be aware that we will mainly take into consideration the following evaluation criteria:

- How close your page is to the mockups, with your added design.
- How clean, testable and organized your code is.
- Clean git log with meaningful commits.
- You implemented the business rules correctly.
- Your code is in a git repo, can be cloned and executed locally on any machine.



Paying

Counterparty Name	Amount
1. Counterparty 1	$100
2. Counterparty 2	$100
3. Counterparty 3	$100
4. Counterparty 4	$100
5. Counterparty 5	$100
....	

Receiving

Counterparty Name	Amount
1. Counterparty 1	$100
2. Counterparty 2	$100
3. Counterparty 3	$100
4. Counterparty 4	$100
5. Counterparty 5	$100
....	